Raytheon

RAYTHEON INTEGRATED DEFENSE SYSTEMS

50 Apple Hill Drive

Tewksbury, MA 01876

Composite Combat Identification (CCID)

Common Reasoning Algorithm Development Program

CDRL A005 - CRA Evaluation Report

Date: 1/20/05

Prime Contract: N00014-04-C-0453

Prepared for:

Office of Naval Research

Ballston Tower One

800 North Quincy St

Arlington, VA 22217-5660

Attn: Ed Khoury, PMR-51

UNCLASSIFIED

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **20 JAN 2005**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Composite Combat Identification (CCID) Common Reasoning Algorithm Development Program**		5a. CONTRACT NUMBER **N00014-04-C-0453**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon Integrated Defense Systems, 50 Apple Hill Drive, Tewksbury, MA 01876**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
Ed Khoury, PMR-51, The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**20**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Financial Status - As of 8/22/2004

		CURRENT MONTH	CUMULATIVE
Total Contract Value	at Sell		$ 333,047
Total Funded Value to date	at Sell		$ 333,047
Total Hours		N/A	N/A
Total Labor to date	at Sell	N/A	N/A
Total Material to date	at Sell	N/A	N/A
Total ODC to date	at Sell	N/A	N/A
Purchase Order Commitments	at Sell	N/A	N/A
Total Actuals and Commitments	at Sell	N/A	N/A

Actuals are reported as of Raytheon fiscal month end.

CRA Evaluation Report

January 20, 2005

1 Introduction

This report presents the CRA (the CCID *C*ommon *R*easoning *A*lgorithm) Evaluation activity and its findings for the ONR CRA Development program, for which Raytheon is the prime contractor and HRL is a subcontractor responsible for the evaluation task. The CRA is a new implementation of the CCID algorithm called CIDER developed by Raytheon/HRL under ONR CCID Phase-I program. Like CIDER, CRA also contains an ingest module and a Dempster-Shafer (D-S) engine. Functionally, the new CRA ingest module replaces the CIDER ingest module, and the new CRA D-S engine replaces HRL's VBS Kernel D-S engine used in CCID Phase-I. The two parts of the system use the same interface protocol and are interchangeable.

The goal of the evaluation task is to evaluate the performance of the new CRA (ingest module and D-S engine) to verify that the new CRA matches or out-performs CIDER.

The remainder of this report is organized as follows:

2 Summary of the Evaluation Results

The evaluation of the CRA is conducted conceptually in two parts, the evaluation of the CRA D-S engine, and the evaluation of the new ingest module. For the evaluation of the CRA D-S engine, we use the original ingest module from CIDER ported to Unix platform to ensure the output from the ingest module and hence input to the CRA D-S engine are the same as that for the VBS Kernel. We were able to verify that the CRA D-S engine produces exactly the same results as those from the VBS Kernel, except for some difference due to random ordering, which does not affect algorithm performance for all practical purposes.

After verifying that the CRA D-S engine is a functional replacement for the HRL VBS Kernel prototype, we evaluated the CRA ingest module by running the CRA ingest module and the CRA D-S engine together. We found that when running using the same configuration and same knowledge base file, CRA matches CIDER exactly with identical results as those from the VBS Kernel, except for some difference due to random ordering, which does not affect algorithm performance for all practical purposes. However, CRA ingest module contains bug fix to the original CIDER code. Formatting errors in the ID knowledge base file was also discovered and fixed. Thus the results produced by the new CRA operating in its normal configuration will produce slightly different numerical results with comparable performance to that of CIDER.

Due to the use of advanced algorithms and code design for the CRA D-S engine, we also see a significant reduction in CPU time and memory usage the CRA D-S engine achieved as compared to the CIDER implementation.

Overall, we conclude that the CRA is a functional equivalent implementation of the CIDER engine, only it is much fast and more memory efficient.

3 Scope of the Evaluation

The scope of this evaluation was to answer the following questions:

- Is the new CRA functionally equivalent to CIDER developed under CCID Phase-I program?
- Is the new CRA able to achieve the same (or better) CID performance as CIDER, according to the metrics defined by ONR during CCID Phase-I program?

The first question was easy to answer, as the new CRA follows the same design and algorithm of CIDER, and uses exactly the same type of support files and interface protocols. Therefore if we achieved the same CID performance, we would have reached a positive conclusion. Therefore the effort for the evaluation was mainly focused on the performance.

In terms of performance, this can be broken down into CID performance and run-time performance. The CID performance can be measured by a set of performance metrics defined by ONR. For run-time, we evaluate CRA's CPU time, memory, and processor usage.

4 Overview of Evaluation Methods

4.1 Evaluation Configurations

Figure 1 shows the software configuration for the CRA evaluation. The CRA consists of the Ingest Module and the Demspter-Shafer (D-S) engine; both are developed under the ONR CRA Development program. For comparison purpose, we also ran the same software configuration as shown in Figure 1, but with the ingest module substituted by the CIDER ingest module developed under the CCID Phase-I. This way, we could focus our performance evaluation on each component of the new CRA.

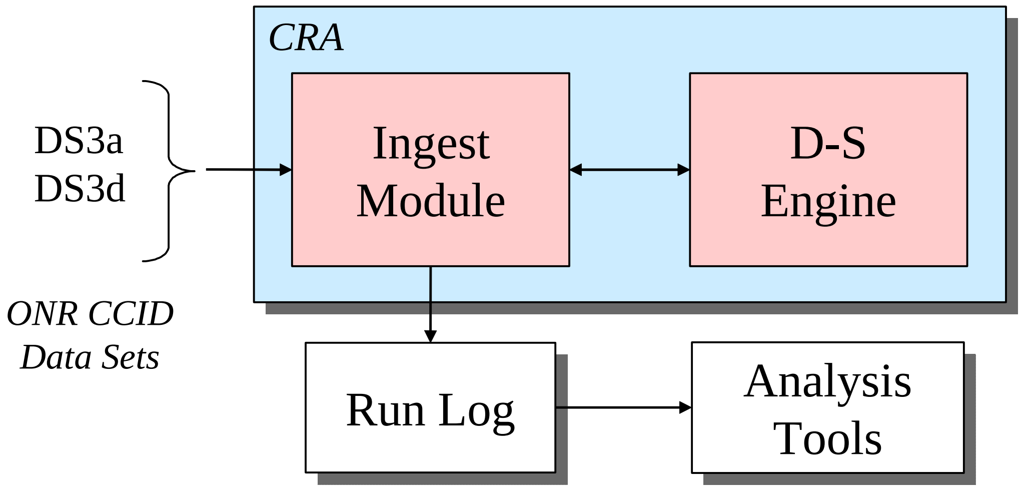


Figure 1. Block Diagram for CRA Evaluation Configuration

For CRA evaluation, we used the original data sets (DS3a & DS3d) as provided by ONR during the ONR CCID Phase-I program. DS3d is the “test” set for CCID Phase-I and so can be used to check the CID performance of the new CRA. DS3a is the most data intensive set, and so is good for evaluating and comparing run-time performance.

The ID knowledge base and other supporting configuration files used for evaluation were the same ones used during the CCID Phase-I. We collected the run logs, and used the performance analysis tools (in Matlab) which were developed during CCID Phase-I. We also developed additional tools to make direct log to log comparisons.

4.2 Hardware and Software Environment

All evaluation runs were conducted on a SunBlade-1000 workstation from Sun Microsystems, with twin UltraSparc-III processors with 8MB external cache and 1GB of RAM running at 750MHz.

The Sun workstation runs Solaris 8 (SunOS 5.8), which is compatible with Solaris 7 for our evaluation purposes.

All software modules were compiled using GNU gcc/g++ version 3.4.2, except for HRL’s VBS Kernel, which was compiled using Sun WorkShop “C” compiler version 6 update 1.

4.3 Baseline Algorithm: CIDER

The evaluation of CRA was driven by the requirements of the CRA Development program, which were to match or do better in CID performance and accuracy than achieved by CIDER as applied to the CCID Phase-I program. Therefore, our baseline software is a version of the CIDER code (CIDER Ingest Module + HRL’s VBS Kernel) ported to Unix (Solaris) platform. The support files (run-time configuration, ID knowledge base, VBS network, and prior belief files) are the same as those used under CCID Phase-I, except for minor reformatting to accommodate the differences between Windows/DOS text files and those under Unix/Solaris.

4.4 Evaluation Methods

Our general evaluation procedure included the following steps:

1. Conduct test runs (including CRA and other configurations for comparison, see Section 4.5) on the evaluation workstation;
2. Log results, run-time, memory, and processor utilization;
3. Run Matlab performance analysis tools, and compare the performance metrics;
4. Compare logs between different runs (baseline and new CRA runs), and collect statistics of any difference.

In the following sub-sections, we give detailed description of each of the above evaluation methods as a reference. First time readers may wish to skip the rest of this section and proceed directly to the next section, and come back to this section for explanation of the different evaluation methods.

4.4.1 CID performance comparison

CID performance are carried out using the performance metrics defined by ONR for CCID Phase-I. These metrics are defined below in this section. The performance metrics are computed for a given declaration threshold from the test run log files using a set of Matlab tools developed in CCID Phase-I. Additional metrics that complements those defined by ONR are also introduced in CCID Phase-I (e.g., error rate) used in plotting the ROC-like curves for the CID performance.

There performance metrics defined by ONR for CCID Phase-I program are:

Accuracy = Q/N (also called Probability of Correct ID)

Reliability = Q/M

Completeness = M/N

where N is the track population, M is the number of declarations made, and Q is the number of declarations that are correct, and $N \geq M \geq Q$. Note that the accuracy and completeness measures penalize inaction. That is, if an algorithm is conservative in making a declaration, it will not achieve better accuracy nor completeness. However, reliability does not take into account of the track population, and only look at the portion of the tracks that are being declared. Undeclared tracks have an ID state of PENDING.

In addition, we define an error measure to complement the above metrics:

Error = (M-Q)/N (also called Probability of False ID)

An ROC curve can be drawn by plotting Accuracy as a function of Error, as we vary some system parameters, in our case, the decision threshold.

For the data sets for evaluation, only Target Type and Nationality truth are given. Therefore all evaluations are based on the metrics for these two ID elements.

4.4.2 Log comparison

For run log comparisons, our primary purpose was to check if the CRA achieves literally exactly the same results as CIDER did during Phase-I CCID program. In most situations, this is only possible if we use the same software and hardware environments, as different machines have different hardware architectures which may affect accuracy of numerical computation. Different compilers also may result in different computation sequence for the same task, potentially introducing further numerical differences.

To solve this issue, we produce the run logs without imposing declaration thresholds on the (pignistic) probability[1] values from the ID reasoning. Rather, the 2 top ranked (in terms of probability value) recommendations are logged, and both the ID states and the associated probability values are compared between different test runs. Because CIDER and the new CRA logs the output in the same order as the input data, we can compare the run logs line by line, and column by column to see if there are any differences. We distinguish and report several different situations where the *same* line of the log file corresponding to the same input from two different test runs might be different:

Case A: The two log files contain the same two top candidates, and their probabilities are equal as well. However, the first candidate in one log file appear as the second candidate in the other log file, therefore is considered different. This case shows up because the two top candidates achieved the same probability in both test runs, but were ordered differently randomly (we call it "random swap") due to platform/compiler differences or algorithm implementation. These situations do not represent algorithm functional difference, and we should consider them as "no difference". In fact, when a proper declaration threshold is used, these cases will fall into "no declaration" or pending, and the two test runs will then be considered to have reached the same results. An example of Case A is shown in the following table:

Case A example	1st candidate ID state	1st candidate probability	2nd candidate ID state	2nd candidate probability
Log1	FRIEND	0.33	FOE	0.33
Log2	FOE	0.33	FRIEND	0.33

Case B: The two log files differ in the 1st candidate's ID state, but it is not Case A. In other words, the first candidate in the two log files are different in their ID state, but it is not due to random swap when the probabilities of the top two candidates are equal. This case is a real difference because if the probability values for the first candidate exceed the set threshold, the declared ID from the two test runs will be different. However, if the probability values are below the threshold, the declared ID will still be the same (i.e., pending). Therefore this may or may not affect the results for a specific threshold. Nevertheless, this situation shows that the two

[1] The CIDER algorithm converts the Demspter-Shafer belief function into a probability form using a transform called "pignistic" transform, which results in a pignistic probability. See [1] for more.

different runs reached different ID states, therefore should raise a **warning**. The following table shows such an example.

Case B example	1st candidate ID state	1st candidate probability	2nd candidate ID state	2nd candidate probability
Log1	FRIEND	0.5	FOE	0.2
Log2	FOE	0.35	FRIEND	0.35

In this example, the top two candidates from the two log files get swapped, but the first candidates' probabilities are different. The declared ID state could be different when a threshold is applied.

Case C: The two log files differ only in the 2nd candidate's ID state, while their first candidates match. This case should be considered as "no difference" as it will always result in the same declaration since the difference is in the 2nd candidate. The cause of the difference in the 2nd candidate is most probably due to the random swap between the 2nd and the 3rd candidates. This case is only for curiosity as it may give us some clue of the inner workings of different algorithms. The following table shows an example of Case C.

Case C example	1st candidate ID state	1st candidate probability	2nd candidate ID state	2nd candidate probability
Log1	FRIEND	0.5	FOE	0.25
Log2	FRIEND	0.5	NEUTRAL	0.25

In summary, for log comparisons we are most concerned with the occurrence of Case B, which indicates potentially true differences in the reasoning results. For comparing the differences of numerical values, we consider any differences in absolute value small than or equal to 1.0-E05 as no difference.

4.4.3 Run-time, memory and processor utilization

For run-time related measurements, since the evaluation workstation contains two CPUs, the ingest module and the D-S engine tend to run on different CPUs concurrently. Our evaluation on run-time performance (CPU time, memory and processor usage) was based on the D-S engine, and not the ingest module because D-S engine is much more time and memory intensive compared with the ingest module.

Run-time figures for the evaluation are obtained using two methods, Unix built-in function "time" and Unix accounting logs. These two approaches gives very similar results, therefore we do not distinguish them in this report.

4.5 Test Runs for the Evaluation

Table 1 shows a summary of four different test run configurations (simply referred to as "test runs" for short here after) conducted during CRA evaluation. Each test run was also carried out using DS3a and DS3d data sets. For "BAA Phase-I" we took the run logs from CCID Phase-I program, and used them to compare to the CIDER ported to Unix. This allowed us to "calibrate" the performance comparisons and log differences between CIDER and the new CRA. For example, we wanted to make sure that the ported code of CIDER achieved the same results as BAA Phase-I, but the run logs maybe slightly different. We further compared the run logs and analyzed the differences to ensure any differences in the results were truly irrelevant to CID performance.

Table 1. Summary of test runs conducted during CRA evaluation

Test Runs	Description
BAA Phase-I	These are not actual test run during the evaluation. Rather the run logs from CCID Phase-I program were used to compare with other test runs. These logs were produced on a Windows 2000 platform during CCID Phase-I program in 2002.
CIDER	Test runs using the original CCID reasoning engine developed under CCID Phase-I, and ported to Unix/Solaris environment.
CRA1	Test runs using CIDER ingest module + the new CRA D-S engine
CRA2	Test runs using the new CRA ingest module + the new CRA D-S engine. It also contains some bug fixes (see Section 5.2.3 for details).

5 CID Performance Evaluation

The purpose of performance evaluation was to verify that under the same conditions and declaration thresholds, CRA produces the same or equivalent performance as CIDER.

We conducted CID performance evaluation by evaluating the performance metrics for all test runs first, then plotted and compared the ROC curves. We describe the results of these activities in the following sections.

5.1 Performance Metrics

The performance metrics were evaluated for each test run against the data sets under a set of declaration thresholds, which was the same as that used in CCID Phase-I for the respective data sets.

Table 2 shows the performance metrics achieved from all test runs for DS3a and DS3d All test runs which achieved the same performance metric values are grouped in the same row.

Table 2. Performance metrics achieved for all test runs.

Track Stat. / Data	Thresholds		Accuracy		Reliability		Completeness		Test Runs used
	Nation	Type	Nation	Type	Nation	Type	Nation	Type	
DS3a	0.55	0.55	0.709	0.541	0.963	0.899	0.736	0.601	CRA1, CIDER, BAA Phase-I
			0.703	0.547	0.963	0.910	0.730	0.601	CRA2
DS3d	0.57	0.60	0.559	0.354	0.888	0.938	0.630	0.378	CRA1&2, CIDER, BAA Phase-I

Table 3. The raw performance data used to compute the metrics shown in Table 2 for all rest runs

Track Stat. / Data	Total Tracks	Total Declared		Total Correct		Test Runs used
		Nation	Type	Nation	Type	
DS3a	148	109	89	105	80	CRA1, CIDER, BAA Phase-I
		108	89	104	81	CRA2
DS3d	127	80	47	71	45	CRA1&2, CIDER, BAA Phase-I

Table 3 shows the raw performance numbers achieved which were used for calculating the metric values shown in Table 2 as defined in Section 4.4.1. For example, accuracy (or probability of correct ID) for Nationality on DS3d in Table 2 is calculated by dividing the total correct declaration (71) by the total number of tracks (127): 71/127=0.559.

From the results presented in Table 2 and Table 3, we can see that CRA1 (testing the CRA D-S engine) and CRA2 (testing the combined CRA ingest module and D-S engine) matches the results of CIDER from BAA Phase-I on DS3d. For DS3a, CRA2 produces slightly different results (slightly better in Target Type, and slightly worse in Nationality) than other test runs. We can see the differences are due to 1 out of over 100 tracks from Table 3. This is due to the differences in numerical results as will be discussed further later on in Section 5.2.3.

5.2 Performance ROC Curves

The tables of metrics shown in the last section only represent the performance of the algorithms when the thresholds are set to specific values for Nationality and Target Type according Table 2. To see a comparison of the algorithm across different threshold settings, we analyze ROC curves.

5.2.1 ROC Basics

We define an ROC curve to be Accuracy as a function of Error. To be consistent with the common definition of an ROC curve, we label the y axis for Accuracy as “Prob. of Correct ID”, and the x axis for Error as “Prob. of False ID”. Note that these two metrics both have as their basis the total population of tracks (see Section 4.4.1).

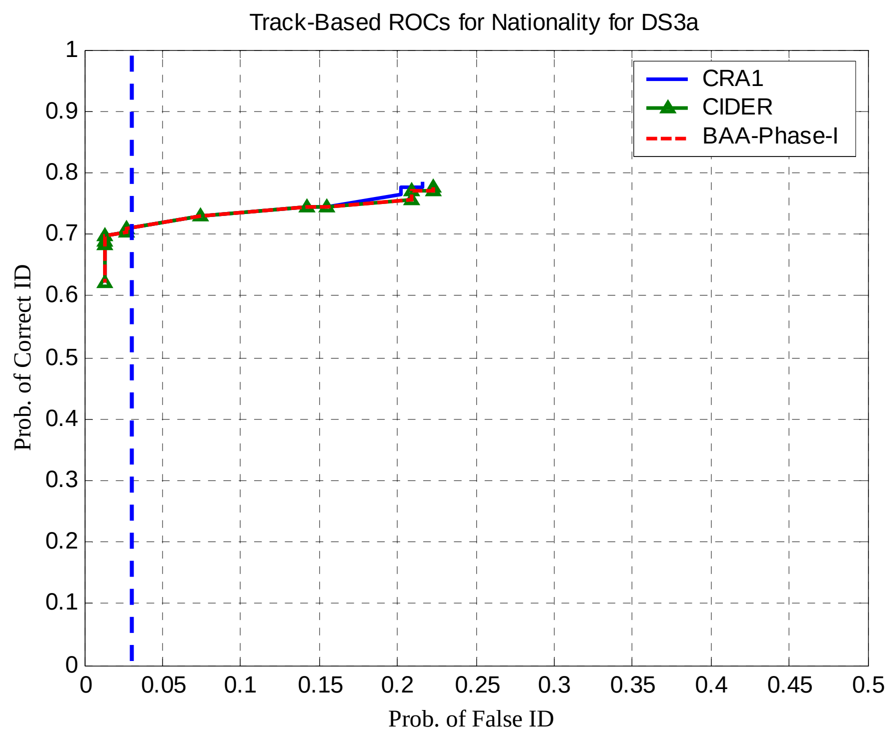
The ROCs are obtained by setting the ID declaration thresholds at different values and then evaluating the resulting performance metrics. Therefore each point on the ROC curve represents an operating point corresponding to a certain threshold. Therefore an ROC curve represents the system performance at various threshold values. When comparing two ROCs, the higher and closer to the left the ROC is, the better the system performance, as the higher curve achieves better correct ID probability at the same false ID probability than the lower one, and the curve closer to the left achieves lower false ID probability at the same correct ID probability.

5.2.2 ROCs for the New D-S Engine

In the first set of comparisons, we plotted the results of test runs CRA1, CIDER, and BAA Phase-I. Recall that CRA1 uses the ingest module of CIDER. So this comparison can tell us whether the new D-S engine matches CIDER. The ROCs for comparison are shown in Figure 2 and Figure 3 for DS3a and DS3d, respectively. The ROC curves for Nationality and Target Type are shown in separate plots.

As can be expected, the ROCs for CIDER and BAA Phase-I match exactly in all cases. CRA1 matches CIDER/BAA Phase-I in most areas across the entire ROCs and only differ in the tail ends (to the right side) of the ROCs for Nationality on DS3a and Target Type on DS3d. The tail end of an ROC corresponds to high false ID (error) rates, and should be avoided during normal operation. Detailed analysis of the run logs suggest that the differences in the ROC curves in the tail end are due to random swapping of the 1st and 2nd candidates in the output when the 1st and 2nd candidates have equal probability values (see Section 4.4.2).

Also note the dashed vertical bars in the ROC plots. The intersection points of the vertical bars and the ROCs represent the operating points chosen for the performance metric evaluation in Section 5.1. The intersection points should match the metric values shown in Table 2. For example, in Figure 2 the vertical dashed bar and the ROCs for Nationality intersect at (0.027, 0.709). In Table 2, the accuracy under Nationality for DS3a (first row) is 0.709, and the completeness under Nationality is 0.736. Therefore 0.736-0.709=0.027, which is probability of false ID (or error rate).



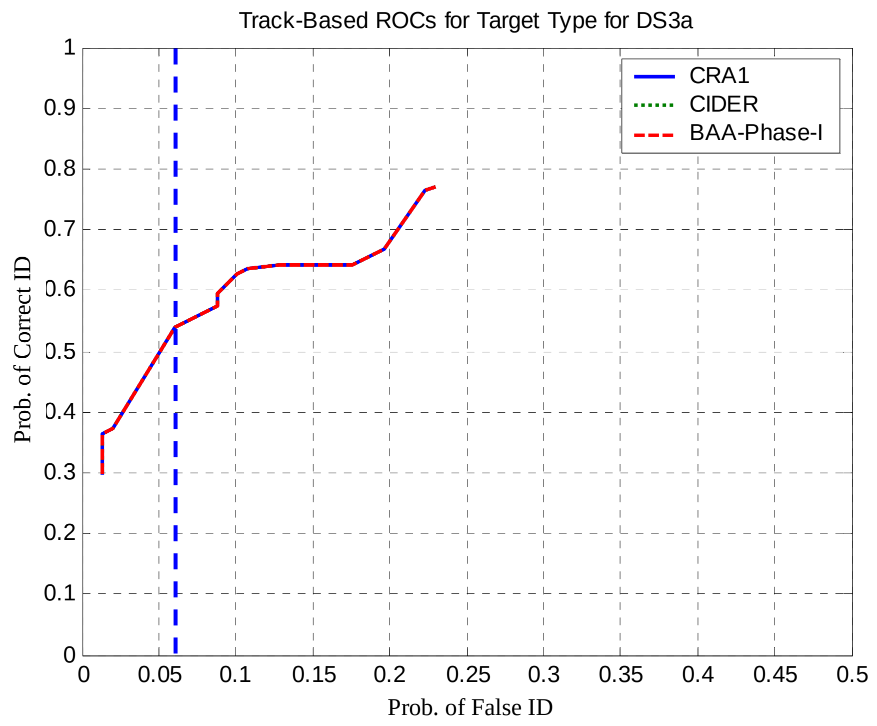
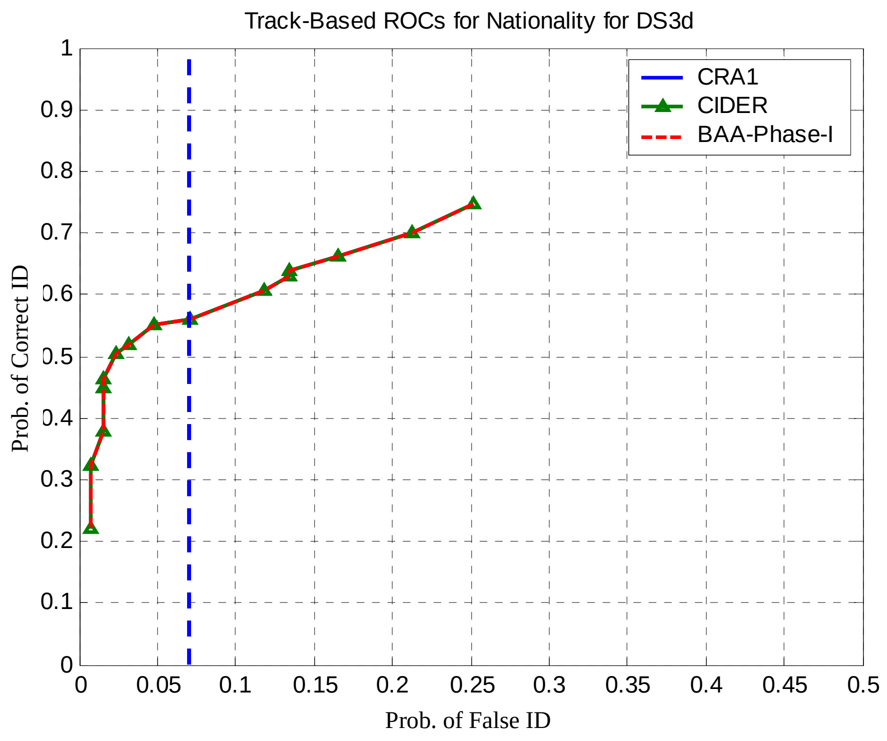


Figure 2. ROC comparisons for DS3a for Nationality and Type for different test runs. The dashed vertical bars in the figure indicate the corresponding operating points for the thresholds set according to those shown in Table 2.



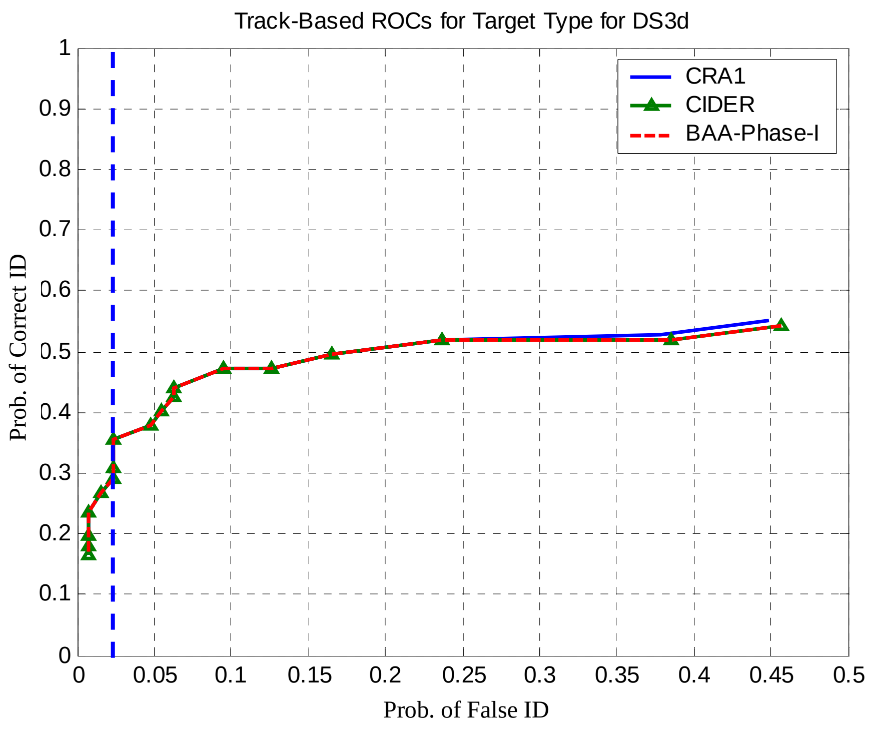


Figure 3. ROC comparisons for DS3d for Nationality and Type for different test runs. The dashed vertical bars in the figure indicate the corresponding operating points for the thresholds set according to those shown in Table 2.

5.2.3 ROCs Using the New Ingest Module

Test run CRA2 uses the new D-S engine and the new ingest module. In the last section, we have shown that the new D-S engine with CIDER ingest module (i.e., CRA1) matches in performance with CIDER/BAA Phase-I, test run CRA2 will test the performance of the new ingest module.

Figure 4 shows the ROCs for CRA2 and CIDER test runs using DS3d. The performance of CRA2 some times lags that of CIDER and sometimes exceeds it. However, overall the differences between the ROCs are minor, and the overall performance of the two should be considered comparable, with CRA2 gaining in areas with low false ID probability, but loosing in areas when the false ID probability is higher.

Some of the differences can be traced back to the new Ingest Module which fixes some problems/bugs that exist in CIDER or its set-up. For example, the ID knowledge base file contains a formatting error in a parameter’s range, which caused the CIDER ingest module to not having been handling that parameter according to the original algorithm design. In another example, the CIDER ingest module was not normalizing the confusion matrix values before some comparison was made, as outlined in the CCID Phase-I final report [1]. The new Ingest Module fixed both these problems, and therefore we expect it to produce slightly different results, especially in the actually probability values from the reasoning due to changes in belief updates.

To be sure we have the new CRA ingest module coded correctly, two special test runs based on the 2 of the 4 shown in Table 1 were conducted. CRA2-NoNorm is the same as CRA2, but with a switch turned on to mimic CIDER’s bug of not doing the needed normalization. CIDER-KB is the same as CIDER but using a knowledge base file with the formatting error corrected. This way, we can do an apple-to-apple comparison between CRA2-NoNorm and CIDER-KB. These two addition test runs are summarized in the table below.

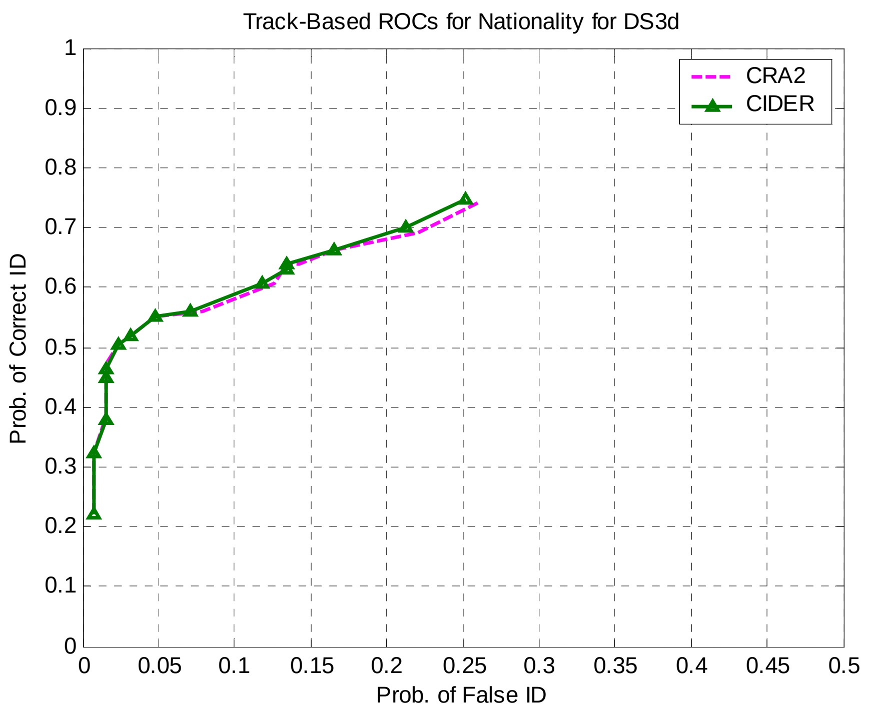
Table 4. New test runs designed to test the equivalence of CRA ingest module to that of CIDER. See text for details.

Test Runs	Description
CIDER-KB	Same as CIDER, but using corrected knowledge base file
CRA2-NoNorm	Same as CRA2, but with the “normalization” turned off

Figure 5 shows the ROC curves for both Nationality and Target Type for these two test runs on DS3d. As can be seen, the ROCs for Nationality match exactly. For Target Type, the curves match exactly on the left side, and only differ slightly in the middle and right side. We will show in the log comparison section (Section 6) that these differences are due to random swap discussed in Section 4.4.2.

5.3 Summary

In this section, through performance metrics and ROC curve comparisons, we have shown that both CRA D-S engine and the ingest module produce either identical or nearly the same performance. The apparent differences in the ROC curves for CRA2 and CIDER test runs were due to changing set-ups and bug fixes in CRA2. Despite of the differences, the declaration performance of CRA2 and CIDER should be considered comparable. We were also able to match the ROC curves when CRA2 and CIDER test runs are conducted under the same condition. This ensures the implementation is correct.



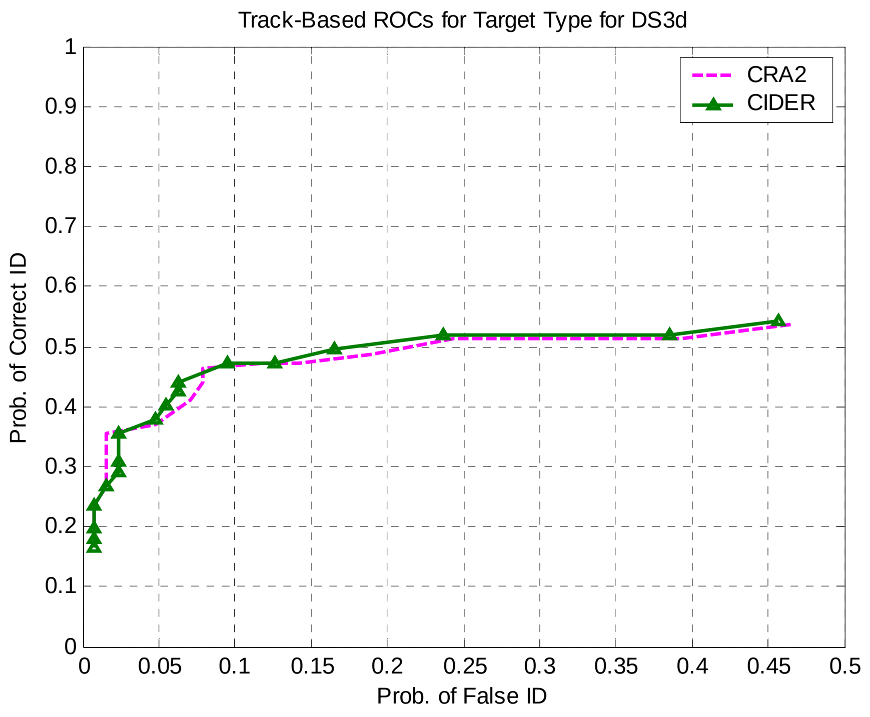
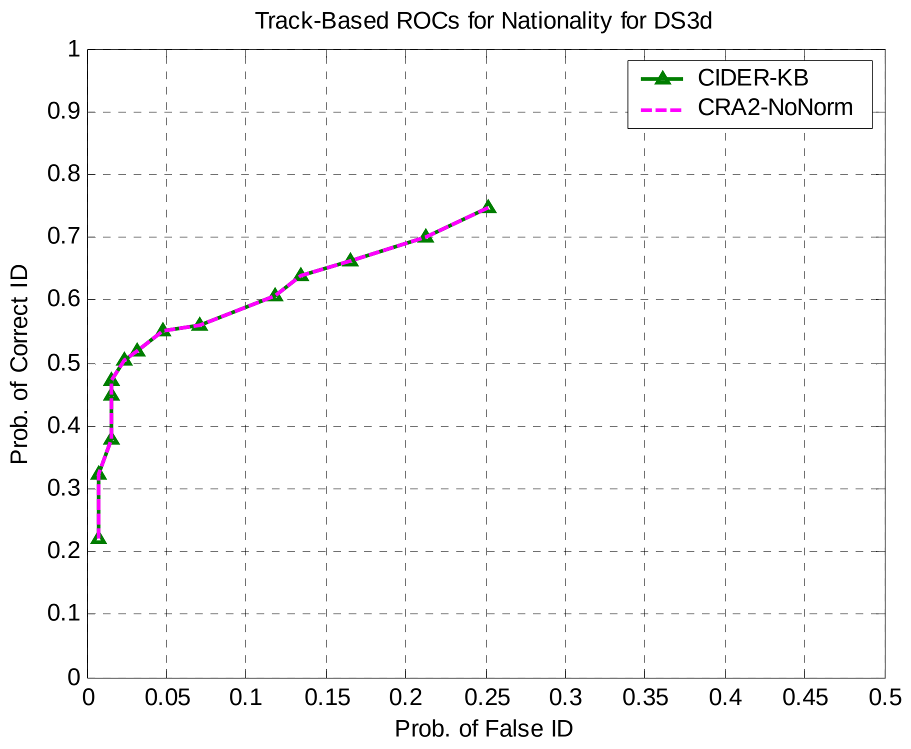


Figure 4. ROCs comparing CRA2 test run with CIDER. Recall that BAA Phase-I has the same ROCs as CIDER (see Figure 2 and Figure 3), and so they are not shown here.



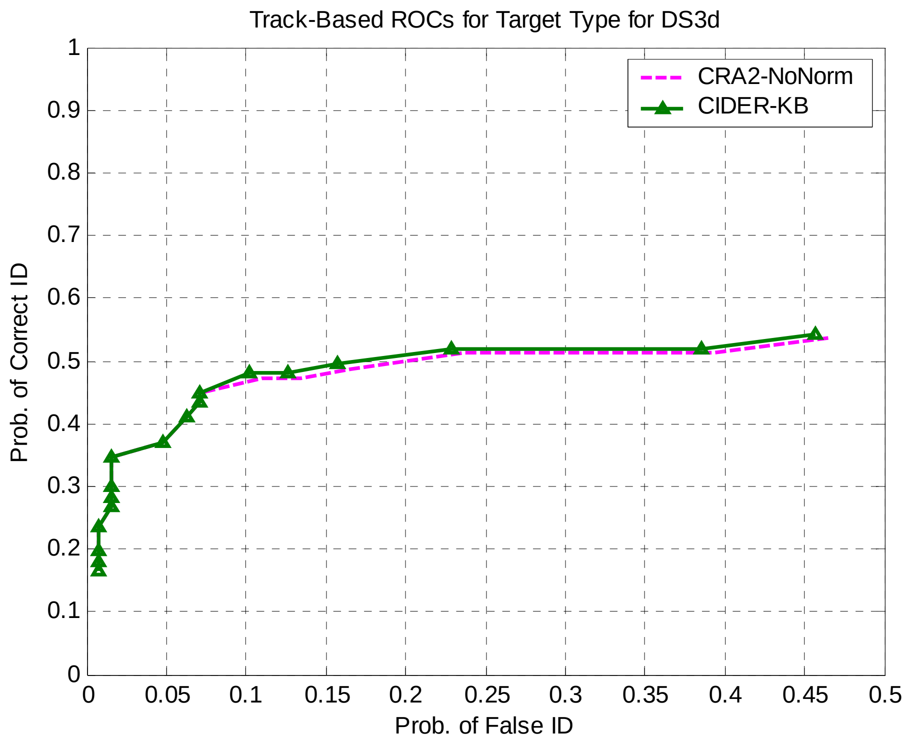


Figure 5. ROC comparisons for test runs CRA2-NoNorm and CIDER-KB. These ROCs shows a match with the results of CIDER and CRA2 under the same condition.

6 Run Log Comparisons

The next step in performance evaluation was to take the actual run logs and compare line by line the differences between the different runs. The purpose of this comparison was to further analyze the differences observed in the performance comparison presented in the last section, and to explain the differences observed in the ROCs. In the following, the findings of log comparisons between pairs of test runs conducted during this evaluation are presented. As discussed in Section 4.4.2, three different cases of log differences were identified. Case A and C are not real differences, while Case B produced different results. For detailed discussion of the methods for run log comparison, please refer to Section 4.4.2.

6.1 CIDER vs. BAA Phase-I

Since CIDER test run is a Unix version of the original CIDER code, we compare it's run log with the original run log "BAA Phase-I" to "calibrate" our comparisons with other test runs for the new CRA. Since the code implementation is the same, any differences between these two log files can be attributed to the compiler differences. The results of comparison are presented in the table below.

Table 5. Results of log comparison between CIDER and BAA Phase-I

Data	Total Lines in the Logs	Case A		Case B		Case C	
		Nation	Type	Nation	Type	Nation	Type
DS3a	7527	0	12	0	0	0	11
DS3d	1693	0	5	0	0	0	0

As can be seen from the above table, there is no difference between the logs except for Case A and C for Target Type. As expected, we see zero differences under the column for Case B. The ROC curves presented in Section 5.2.2 (Figure 2 and Figure 3) also confirmed this observation.

6.2 CRA1 vs. BAA Phase-I

Results of log comparison between CRA1 and BAA Phase-I are shown in Table 6.

Table 6. Results of log comparison between BAA Phase-I and CRA1

Data	Total Lines in the Logs	Case A		Case B		Case C	
		Nation	Type	Nation	Type	Nation	Type
DS3a	7527	331	1	0	0	1171	350
DS3d	1693	14	4	0	0	6	24

Again, note the absence of any occurrence of Case B differences. However, we observed many more Case A and Case B differences between CRA1 and BAA Phase-I. These differences were due to candidate position swapping when the probabilities were equal.

As has been shown in Section 5.2, these differences only affect the tail ends of the ROCs corresponding to high false ID probability, which a normal CID engine designer will avoid. The cause for the candidate position swapping is due to the different implementations between the new CRA D-S engine and VBS Kernel in maintaining the internal list of ID states. Compiler differences may also be a factor, but is a lesser contributor, as can be seen from the results presented in Section 6.1.

6.3 CRA2 vs. BAA Phase-I and CRA2-NoNorm vs. CIDER-KB

For CRA2 test run, the new Ingest Module and the new D-S engine were used. As discussed in Section 5.2.3, due to the facts that the CRA ingest module implementation contained bug fixes and the CRA2 test run uses a format-corrected knowledge base file, the test run results were different in numerical values (the probabilities). The differences in the appearance of the ROCs in Figure 4 demonstrated this clearly. In this situation, a direct comparison of the run logs was no longer meaningful.

Rather than comparing CRA2 and BAA Phase-I logs, we compared the run logs of test runs CRA2-NoNorm and CIDER-KB described in Section 5.2.3, Table 4. These test runs are conducted under the same conditions and therefore helped to demonstrate that the CRA code was implemented correctly according to CIDER’s design. The results are shown in Table 7.

Table 7. Log comparison results for CRA2-NoNorm and CIDER-KB.

Data	Total Lines in the Logs	Case A		Case B		Case C	
		Nation	Type	Nation	Type	Nation	Type
DS3d	1693	3	106	0	0	0	35

Clearly, the absence of Case B difference proves that these two test runs match. The Case A and C differences are the only causes for the slight differences in the ROC shown in Figure 5.

6.4 Summary

Log comparisons show that porting the CIDER code to Unix platform without code/algorithm change introduces few differences in the logs due to random swap that do not affect the performance. The absence of Case B differences in CRA1 and BAA Phase-I test run logs showed that the CRA D-S engine matched the VBS Kernel D-S engine. Although we were able to do a meaningful log comparison for CRA2, we were able to develop special test runs for CRA2 (CRA2-NoNorm) and CIDER (CIDER-KB) that allowed us to run both ingest modules under the same condition. The results proved the CRA ingest module does indeed match the CIDER ingest module. This exercise gave us confidence to say the differences we see in the ROCs between CRA2 and BAA Phase-I are entirely related to the bug fixes and correction in knowledge base formatting.

7 CPU Time, Memory and Processor Usage

Table 8 shows the results of run-time evaluation, with comparable figures from VBS Kernel. For the evaluation CRA D-S engine, we used the CRA1 test run configuration with CIDER ingest module and CRA D-S engine. For VBS Kernel, we used the CIDER test run configuration (see Section 4.5). The CRA ingest module is not included in this part of the evaluation since it takes much less CPU time and memory compared to those taken by either CRA D-S engine or the VBS Kernel.

Table 8. Results for timing, memory and processor usage for the CRA D-S engine. For these results, the test run configuration of "CRA1" was used and only the CRA D-S engine processes was evaluated.

Data	CPU time/Elapsed time (hour:minute:second)		Peak Memory (MB)		Processor Utilization (%)	
	CRA D-S Engine	VBS Kernel	CRA D-S Engine	VBS Kernel	CRA D-S Engine	VBS Kernel
DS3a	0:5:24 / 0:5:54	6:55:50 / 7:11:29	65	159	92	97
DS3d	0:3:2 / 0:3:18	0:12:57 / 0:15:41	96	115	92	83

As can be seen that the CRA D-S engine is much faster than the VBS Kernel, and also more efficient in memory usage. The timing comparison on DS3a is especially dramatic, with CRA D-S engine achieving more than 75 times the speed of that for VBS Kernel.

8 Conclusion

We have described our objectives, approaches and results of the CRA Evaluation task under the CRA Development program. Through careful design and planning of the evaluation process, analysis tools, and the lengthy process of conducting the evaluation tests and result analysis, we can conclude that the new CRA is a functional equivalent of the CIDER engine developed under CCID Phase-I program. The new CRA achieves the same CID performance metric values as CIDER does, and produces equivalent performance as evaluated through various ROC curves and performance metrics. The new CRA D-S engine is also much faster, achieving 4 to 75 times speed up compared to CIDER as evaluated based on DS3a and DS3d data sets from CCID Phase-I program. It also consumes appreciably less memory (sometimes less than an half) than the memory required by VBS Kernel.

Reference

[1] Barbara Blyth and Yang Chen, "ONR CCID Program (BAA-01-024) Final Report" by Raytheon Co. and HRL Laboratories, LLC, October 2002.